K Nearest Neighbour

Aim: To implement a Machine Learning Classification model using a K Nearest Neighbors Classifier algorithm and enhance the model by K Fold and GridSearchCV cross-validation.

In [73]:
```
import numpy as np
import pandas as pd
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
```

In [74]:
```
data = pd.read_csv(r"Practical5.csv")
X = data.iloc[:, [1, 2, 3, 4, 5, 6, 7]].values
y = data.iloc[:, -1].values
```

In [75]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
X[:,0] = le.fit_transform(X[:,0])
```

splitting up the dataset

In [76]:
```
X_train, X_test, y_train, y_test = train_test_split(
            X, y, test_size = 0.2, random_state=42)
```

In [77]:
```
knn = KNeighborsClassifier(n_neighbors=13)
```

Training the model

In [78]:
```
knn.fit(X_train, y_train)
```

Out[78]:
```
KNeighborsClassifier(n_neighbors=13)
```

Making the predictions

In [79]:
```
y_pred=knn.predict(X_test)
print(knn.predict(X_test))
```

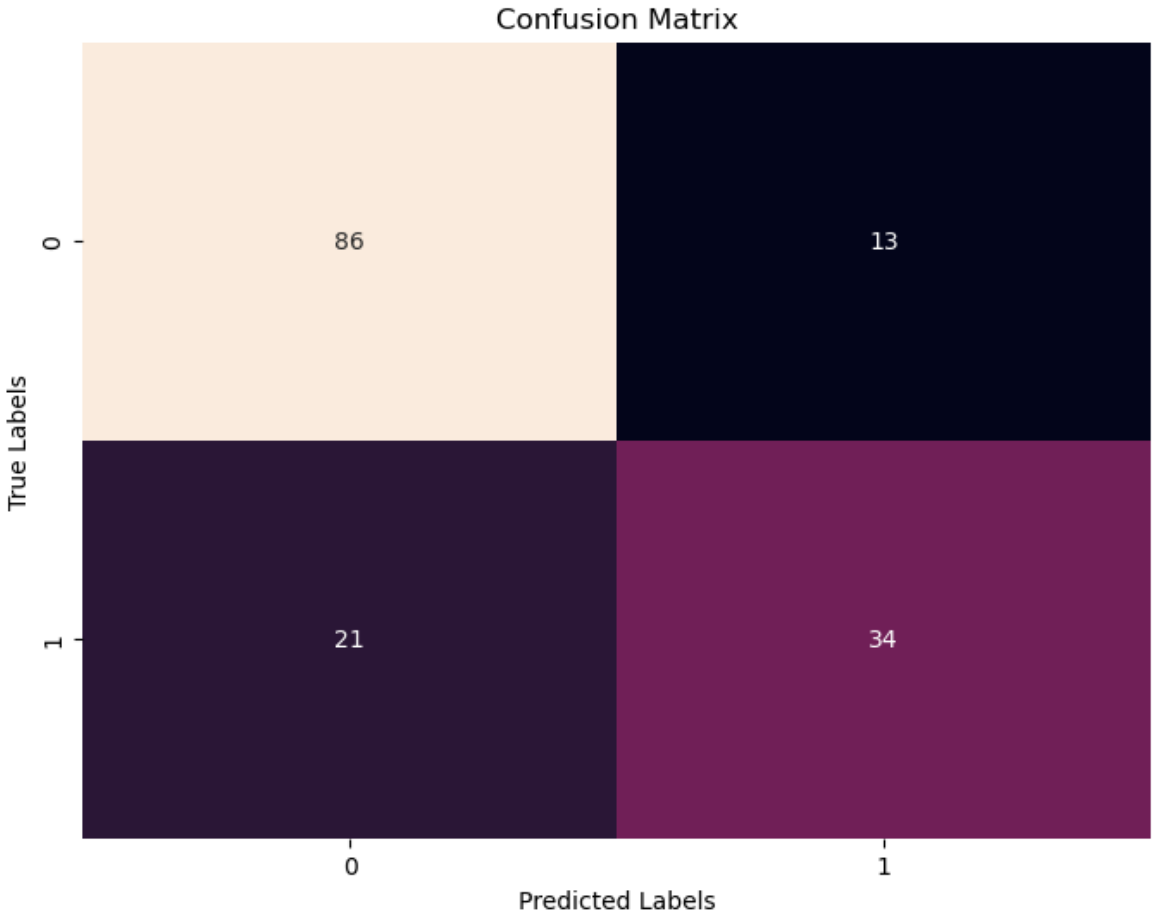
```
[0 0 0 0 1 0 0 0 1 0 1 1 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 1 0 1 1 0 1 1
 0 0 1 0 0 1 1 0 1 0 0 0 1 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 0 0 0 0
 0 1 0 0 0 0 1 0 0 1 0 1 1 0 0 0 0 0 0 1 0 0 1 0 1 0 0 0 0 0 1 0 0 1 0 1 0
 1 0 1 0 0 1 0 0 0 0 0 0 0 0 1 0 1 1 1 1 0 0 1 1 0 0 1 1 0 0 0 0 1 0 0 0 0
 0 1 0 0 0 0]
```

In [80]:
```
from sklearn.metrics import confusion_matrix
cm= confusion_matrix(y_test,y_pred)
```

```
print(cm)
```

```
[[86 13]
 [21 34]]
```

In [81]:
```
import seaborn as sns
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cbar=False)
plt.title('Confusion Matrix')
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.show()
```



KFold cross validation

In [82]:
```
X_train, X_test, y_train, y_test = train_test_split(
            X, y, test_size = 0.2, random_state=1)
```

In [83]:
```
from sklearn.model_selection import KFold
from sklearn.metrics import precision_recall_fscore_support
from sklearn.metrics import confusion_matrix
#Data is splited into 10 same parts
cv = KFold(n_splits=15)
# perform cross-validation procedure
for train_ix, test_ix in cv.split(X):
    # split data
    X_train, X_test = X[train_ix, :], X[test_ix, :]
    y_train, y_test = y[train_ix], y[test_ix]
    # fit and evaluate a model
    knn = KNeighborsClassifier(n_neighbors=7)
```

```
knn.fit(X_train, y_train)
y_pred=knn.predict(X_test)
print(knn.predict(X_test))
#draw confusion matrix
cm= confusion_matrix(y_test,y_pred)
print(cm)
#find metrices of evalution
precision, recall, f1_score,_ = precision_recall_fscore_support(y_test, y_pr
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1_score)
```

```
[0 0 1 0 1 0 0 1 1 0 0 1 0 1 1 0 0 0 0 0 0 0 1 0 1 0 1 0 1 0 1 1 0 0 1 0 0
 0 0 0 0 1 1 1 1 1 1 0 0 0 0 0]
[[19  8]
 [13 12]]
Precision: [0.59375 0.6    ]
Recall: [0.7037037 0.48     ]
F1 Score: [0.6440678  0.53333333]
[0 1 1 0 1 0 1 0 0 0 0 1 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1
 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0]
[[34  5]
 [ 6  7]]
Precision: [0.85       0.58333333]
Recall: [0.87179487 0.53846154]
F1 Score: [0.86075949 0.56      ]
[0 0 0 1 0 0 1 1 0 0 1 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0
 1 0 1 0 0 0 0 1 0 0 0 1 1 1 1]
[[30  4]
 [ 7 11]]
Precision: [0.81081081 0.73333333]
Recall: [0.88235294 0.61111111]
F1 Score: [0.84507042 0.66666667]
[0 0 0 1 1 0 0 0 0 1 0 1 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 1 1 0 0 1 0 0 1
 1 0 0 0 0 0 0 0 0 0 0 1 0 1]
[[27  4]
 [10 10]]
Precision: [0.72972973 0.71428571]
Recall: [0.87096774 0.5       ]
F1 Score: [0.79411765 0.58823529]
[1 0 1 0 0 1 0 0 0 0 0 0 1 1 1 0 1 0 0 0 1 1 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0
 1 1 1 1 0 0 1 0 0 0 0 0 0 0]
[[22  7]
 [11 11]]
Precision: [0.66666667 0.61111111]
Recall: [0.75862069 0.5       ]
F1 Score: [0.70967742 0.55      ]
[1 1 1 1 0 0 0 0 1 0 0 1 0 0 1 0 0 0 0 0 0 0 0 1 1 1 0 1 1 0 0 0 0 0 0 0 0
 1 0 0 0 0 1 0 0 0 0 0 0 1 0 0]
[[22  8]
 [14  7]]
Precision: [0.61111111 0.46666667]
Recall: [0.73333333 0.33333333]
F1 Score: [0.66666667 0.38888889]
[0 0 0 1 0 1 0 0 1 0 1 0 0 0 0 0 0 1 0 1 0 0 1 0 1 0 0 1 1 1 0 1 0 0 0 1 0 0
 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1]
[[26  6]
 [ 9 10]]
Precision: [0.74285714 0.625     ]
Recall: [0.8125     0.52631579]
F1 Score: [0.7761194  0.57142857]
[1 1 1 0 0 0 0 0 0 1 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 1 1 0 0 0 1 0 0 1 0 0
 0 0 1 0 1 0 0 1 1 0 0 1 1 0]
[[26  5]
 [ 8 12]]
Precision: [0.76470588 0.70588235]
Recall: [0.83870968 0.6       ]
F1 Score: [0.8        0.64864865]
[0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 1 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0
 0 0 0 0 0 0 0 1 1 0 1 0 0 0]
[[32  2]
 [ 9  8]]
```

```
Precision: [0.7804878 0.8      ]
Recall: [0.94117647 0.47058824]
F1 Score: [0.85333333 0.59259259]
[0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 1 0 0 0 1 0 1 1 0 1 0 1 0 0 0 1 0 0 1
 1 0 0 0 0 0 0 0 1 0 0 0 0 0]
[[32  9]
 [ 7  3]]
Precision: [0.82051282 0.25      ]
Recall: [0.7804878 0.3      ]
F1 Score: [0.8        0.27272727]
[0 0 0 1 1 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 1 0 1
 0 0 0 0 0 0 0 0 0 0 1 1 0 0]
[[35  4]
 [ 5  7]]
Precision: [0.875      0.63636364]
Recall: [0.8974359  0.58333333]
F1 Score: [0.88607595 0.60869565]
[0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 1 1 0 0 0 1 0 1 0 1 0 1 0 1 0 0 1 0 0 1 0 0
 0 1 1 1 0 1 0 0 0 0 1 1 0 1]
[[30  4]
 [ 2 15]]
Precision: [0.9375     0.78947368]
Recall: [0.88235294 0.88235294]
F1 Score: [0.90909091 0.83333333]
[0 0 0 0 1 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 1 1 1 0 0 0
 0 0 0 1 0 1 0 0 1 1 1 0 1 0]
[[30  6]
 [ 6  9]]
Precision: [0.83333333 0.6       ]
Recall: [0.83333333 0.6       ]
F1 Score: [0.83333333 0.6       ]
[0 0 0 1 1 0 0 1 0 1 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 1 0 1 1 0 0 0 0 0 1
 1 0 0 0 0 1 0 0 0 0 0 0 1 0 1]
[[25  6]
 [10 10]]
Precision: [0.71428571 0.625     ]
Recall: [0.80645161 0.5       ]
F1 Score: [0.75757576 0.55555556]
[0 0 1 0 0 0 0 0 1 1 0 1 0 0 0 1 1 0 0 0 0 0 1 1 0 0 1 1 0 0 0 1 0 0 0 0 1
 1 1 0 1 0 1 0 1 0 1 0 0 1 0]
[[26  6]
 [ 6 13]]
Precision: [0.8125     0.68421053]
Recall: [0.8125     0.68421053]
F1 Score: [0.8125     0.68421053]
```

GridSearchCV

In [84]:

```
from sklearn.model_selection import KFold, GridSearchCV
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix, precision_recall_fscore_support

# Define the KFold cross-validation
cv = KFold(n_splits=12)

param_grid = {'n_neighbors': list(range(1, 21, 2))}
```

```
# Initialize the KNN classifier
knn = KNeighborsClassifier()

# Initialize GridSearchCV
grid_search = GridSearchCV(estimator=knn, param_grid=param_grid, cv=cv, scoring=

# Perform cross-validation procedure with GridSearchCV
for train_ix, test_ix in cv.split(X):
    # Split data
    X_train, X_test = X[train_ix, :], X[test_ix, :]
    y_train, y_test = y[train_ix], y[test_ix]

    # Fit model using GridSearchCV
    grid_search.fit(X_train, y_train)

    # Get the best KNN model found by GridSearchCV
    best_knn = grid_search.best_estimator_

    # Predict
    y_pred = best_knn.predict(X_test)

    # Evaluate
    cm = confusion_matrix(y_test, y_pred)
    precision, recall, f1_score, _ = precision_recall_fscore_support(y_test, y_p

    # You can also access the best hyperparameters found
print("Best parameters found by GridSearchCV:", grid_search.best_params_)
```

```
Best parameters found by GridSearchCV: {'n_neighbors': 17}
```

In [85]:
```
print("Best parameters found by GridSearchCV:", grid_search.best_params_)
```

```
Best parameters found by GridSearchCV: {'n_neighbors': 17}
```

Building model for the best parameter

In [86]:
```
knn = KNeighborsClassifier(n_neighbors=17)
```

In [87]:
```
knn.fit(X_train, y_train)
```

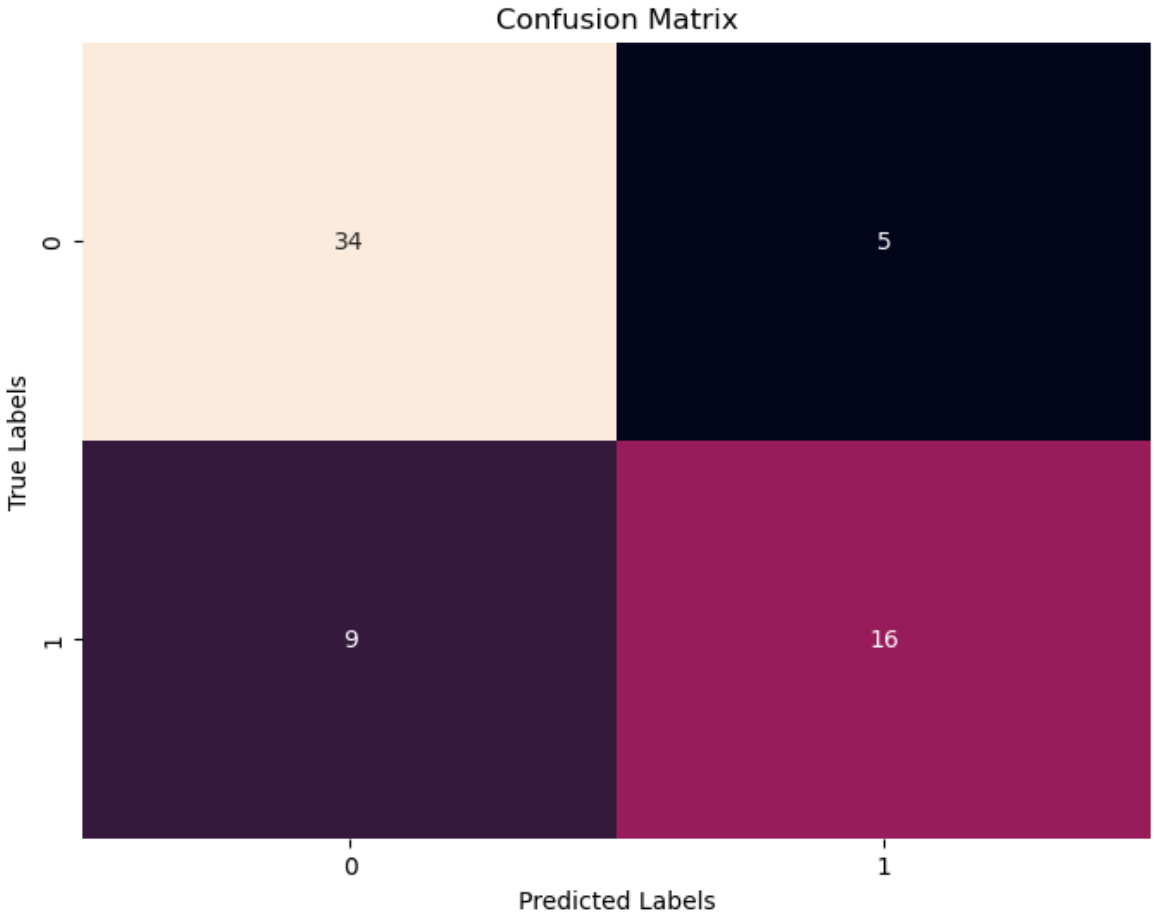
Out[87]:
```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=17)
```

In [88]:
```
y_pred=knn.predict(X_test)
```

In [89]:
```
cm= confusion_matrix(y_test,y_pred)
print(cm)
```

```
[[34  5]
 [ 9 16]]
```

In [90]:
```
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cbar=False)
plt.title('Confusion Matrix')
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.show()
```


Confusion Matrix
34
5
9
16
0
1
True Labels
0
1
Predicted Labels